

### REMARKS

Claims 9-12 and 23 are pending in the present application. Reconsideration of the claims is respectfully requested.

#### **I. Obviousness-type Double Patenting**

The Examiner has rejected claims 9-12 and 23 under the judicially-created doctrine of obviousness-type double patenting as being unpatentable in view of claims 7-17 of U.S. Patent No. 6,442,685, which was filed on the same date as the present application. Applicants have submitted herewith a terminal disclaimer under 37 CFR § 1.321, thus obviating the double patenting rejection. Applicants respectfully request that the rejection be withdrawn.

#### **II. 35 U.S.C. § 103, Obviousness, Claims 9-12 and 23**

The Examiner has rejected claims 9-12 and 23 under 35 U.S.C. § 103 as being unpatentable over *Kandasamy* (US Patent No. 6,219,799) in view of *Chrabaszcz* (US Patent No. 6,134,673). This rejection is respectfully traversed.

With regard to claim 9, the Examiner stated:

As per claim 9, Kandasamy discloses the invention substantially as claimed including a method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:

dynamically modifying a first set of server names for a first server by adding a first server name to the first set of server names [Abstract; col. 2, lines 27-32 and 59-67];

dynamically modifying a second set of server names for a second server by adding a second server name to the second set of server names [Abstract; col. 2, lines 27-32 and 59-67];

determining that the first server requires reconfiguration [Abstract; and col. 3, lines 15-16];

dynamically modifying a first set of server names by adding the second server name to the first set of server names [col. 3, lines 5-8 and 15-23; claim 1].

Kandasamy does not disclose

the first server responds to requests directed to the first set of server names; and

the second server responds to requests directed to the second set of server names.

Chrabaszcz discloses

the first server responds to requests directed to the first set of server names [col. 6, lines 16-18 and 39-56]; and  
the second server responds to requests directed to the second set of server names [col. 6, lines 16-18 and 39-56].

It would have been obvious to a person skill in the art at the time the invention was made to combine the teaching of Kandasamy and Chrabaszcz because Chrabaszcz's teaching would provide multiple services from difference servers for clients which increases scalability.

(Office Action, dated April 18, 2003, pages 4-5).

For an invention to be prima facie obvious, the prior art must teach or suggest all claim limitations. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). Independent claim 9, which is representative of independent claims 11 and 23, reads as follows:

9. A method for reconfiguring servers in a distributed data processing system, the method comprising the computer-implemented steps of:  
dynamically modifying a first set of server names for a first server by adding a first server name to the first set of server names, wherein the first server responds to requests directed to the first set of server names;  
dynamically modifying a second set of server names for a second server by adding a second server name to the second set of server names, wherein the second server responds to requests directed to the second set of server names;  
determining that the first server requires reconfiguration; and  
dynamically modifying the first set of server names by adding the second server name to the first set of server names.

Claim 9 recites dynamically modifying a first set of server names for a first server *by adding a first server name to the first set of server names*, dynamically modifying a second set of server names for a second server *by adding a second server name to the second set of server names*, and dynamically modifying the first set of server names *by adding the second server name to the first set of server names*.

Neither *Kandasamy* nor *Chrabaszcz*, as cited by the Examiner, teach or suggest these steps. Although the Examiner is correct in noting that *Kandasamy* teaches supporting multiple names in a single computer, the method of providing this support in *Kandasamy* differs from the method as recited in claim 9. For example, *Kandasamy* does not teach or suggest the features of adding a first server name to the first set of server names, adding a second server name to the second set of server names, or adding the second server name to the first set of server names. Instead, *Kandasamy* teaches

supporting multiple node-names by using a special server name “\*SMBSERVER”. *Kandasamy* teaches having the SMB server listen for network requests on a computer name (see col. 2, lines 63-64) and having the Name Proxy Driver (“NPD”) replace the called name with “\*SMBSERVER” (see col. 4, lines 45-49). Thus, contrary to the Examiner’s assertion, *Kandasamy* does not teach the features of adding a first server name to the first set of server names, adding a second server name to the second set of server names, or adding the second server name to the first set of server names. *Kandasamy* merely uses a proxy driver to replace the called name rather than add the name to a set of server names.

*Chrabaszcz* does not cure the deficiencies of *Kandasamy*. *Chrabaszcz* fails to teach or suggest the desirability of adding a first server name to the first set of server names, adding a second server name to the second set of server names, or adding the second server name to the first set of server names. In fact, there is no mention in *Chrabaszcz* of the problem of supporting multiple names in a single computer.

Therefore, even if *Kandasamy* could be combined with *Chrabaszcz*, the presently claimed invention would not be reached because the features of adding a first server name to the first set of server names, adding a second server name to the second set of server names, or adding the second server name to the first set of server names are missing from both prior art references.

Furthermore, a proper prima facie case of obviousness cannot be established by combining the teachings of the prior art absent some teaching, incentive, or suggestion supporting the combination. *In re Napier*, 55 F.3d 610, 613, 34 U.S.P.Q.2d 1782, 1784 (Fed. Cir. 1995); *In re Bond*, 910 F.2d 831, 834, 15 U.S.P.Q.2d 1566, 1568 (Fed. Cir. 1990). In making an obviousness determination, one cannot pick and choose among the individual elements or assorted prior art references to recreate the claimed invention. *Symbol Technologies, Inc. v. Opticon, Inc.*, 935 F.2d 1569, 19 U.S.P.Q. 2d 1241 (Fed. Cir. 1991). Instead, whether the prior art made obvious the invention must be determined by looking for some teaching or suggestion in the references to support their use in the particular claimed invention. *Id.*

Claim 9 recites dynamically modifying a first set of server names for a first server by adding a first server name to the first set of server names, wherein the first

server responds to requests directed to the first set of server names. Applicants agree with the Examiner's statement that *Kandasamy* does not teach first server responds to requests directed to the first set of server names (*Office Action*, page 4).

Although *Chrabaszcz* discloses having a user access a server by means of a user workstation connected to a LAN line (col. 6, lines 16-18), one of ordinary skill in the art would not combine *Kandasamy* with *Chrabaszcz* when the references are considered as a whole. In considering a reference as a whole, one of ordinary skill in the art would take into account the problems recognized and solved by the reference. For example, *Chrabaszcz* teaches:

A method for fault tolerant execution of an application program, in a server network having a first and second server, wherein the method includes: executing the application program in the first server; storing an object which represents the program in a cluster network databasc, wherein the object contains information pertaining to the program; detecting a failure of the first server; and executing the application program in the second server upon detection of the failure of the first server, in accordance with the information in the object. The information may include: a host server attribute which identifies which server is currently executing the program; a primary server attribute which identifies which server is primarily responsible for executing the program; and a backup server attribute which identifies which server is a backup server for executing the program if the primary server experiences a failure.

(*Chrabaszcz*, Abstract).

In one embodiment, the invention involves an enhanced network directory databasc which operates in conjunction with server resident processes, i.e., Netframe Cluster software, to remap the execution of clustered applications, or clustered programs, in the event of a server failure. In one embodiment, the enhanced network directory database is replicated throughout all servers of the network. As explained in further detail below, the database stores configuration data ("objects") which contain for each clustered application, a primary and a secondary server affiliation as well as other information. Initially, all users access a clustered application through the server identified in the object as being the primary server for that clustered application.

When server resident processes, otherwise known as Netframe Cluster software, detect a failure of the primary server, the enhanced database is updated to reflect the failure of the primary server, and to change the affiliation of the clustered application from its primary to its secondary, or backup, server. In one embodiment, the updating and

remapping are accomplished by server resident processes which detect a failure of the primary server, and remap the clustered application server affiliation. This remapping occurs transparently to whichever user is accessing the clustered application. Thus, all users access a clustered application through the backup server. This process may be reversed when the primary server resumes operation, the backup server unloads the clustered application from memory, and then users may again access the clustered application through the primary server, thereby regaining fault tolerance, i.e. backup, capability.

No dedicated redundant resources are required to implement the current invention. Rather, the current invention allows server resident processes to intelligently relocate cluster applications to servers in the event of server failure. A server may be a primary server with respect to a clustered application loaded in its memory, a secondary or backup server with respect to another clustered application stored in its hard drive, though not loaded in memory, and function as a fully functional file server.

(Chrabaszcz, col. 5, line 40 to col. 6, line 11). Thus, Chrabaszcz is directed toward relocating cluster applications to a backup server in the event of a primary server failure. Chrabaszcz is concerned with remapping clustered application affiliations from the failed primary server to the backup server, as also illustrated in Figure 3C:

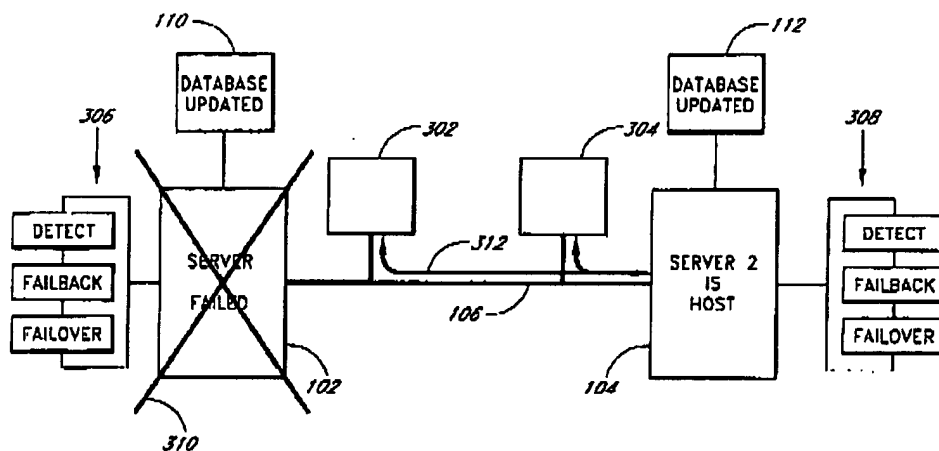


FIG. 3C

In FIG. 3C, the process 308 running on the second server 104 has detected the failure of the first server 102. As described above, the clustered application that is loaded into the RAM of the first server 102 is

represented in the databases 110 and 112 by an object. Since the object contained in databases 110 and 112 designates the second server 104 as the backup server, the second server 104 will load its own copy of the clustered application from its hard drive and execute the clustered application upon detection of the primary server failure. Upon detection of the failure of a server, the Netframe Cluster software updates the database 112. The object in the databases is updated such that the value of the host server attribute is changed to the second server 104, the backup server. Because the attribute values in the object for the cluster application have been changed, communications with the clustered application will now be rerouted through server 104. This process is referred to as the failover process herein.

(*Chrabaszcz*, col. 8, line 64 to col. 14). As is described above, *Chrabaszcz* teaches changing the host server attribute value from the primary server to the backup server if a failure of the primary server is detected. The attribute values in the cluster application object in the enhanced database are changed to name the backup server to reflect failure of the primary server. The backup server then loads its own copy of the clustered application from its hard drive and executes the clustered application. Thus, *Chrabaszcz* teaches changing the host server attribute to name the backup server as the host server of the application. Nowhere does *Chrabaszcz* teach or suggest the desirability of adding the backup server name to a set of server names for the primary server (or vice versa) in order to allow the primary server to support the backup server name in addition to its own. In fact, there is no mention in *Chrabaszcz* of the problem of supporting multiple names in a single computer.

The *Kandasamy* reference is directed toward providing redundant access to a network-based storage medium and supporting multiple node-names therein, where NETBIOS and SMB services are assumed in a Windows NT operating system. Since the NETBIOS and SMB services of the Microsoft Windows NT operating system depend on the "name" of the computer to function, *Kandasamy* is concerned with providing this functionality by having the computer that "takes over" the functions of the failed computer support more than one name (e.g. its own name plus that of the failed computer). Thus, while the *Chrabaszcz* reference is directed toward remapping clustered application affiliations from a failed primary server to the backup server in order to have the backup server execute the application, the *Kandasamy* reference is directed toward

providing support for multiple node-names. In other words, *Chrabaszcz* is merely concerned with rerouting communications to a backup server if the primary server experiences a failure, while *Kandasamy* is concerned with having a single computer support any number of unique computer names so that the computer can take over all the functions of a failed computer in the network. The Examiner has not provided any motivation *from the prior art* to suggest that making all the necessary modifications to the reference teachings to achieve the present invention would be desirable. If the Examiner cannot make such a showing, then the Examiner could only have relied on hindsight with the benefit of Applicants' disclosure to develop an incentive for the changes. The Examiner's hindsight observations, in the absence of additional support in the available art at the time of invention, are insufficient to make the necessary showing of a motivation or incentive to modify or combine reference teachings to achieve the present invention.

Therefore, absent some teaching, suggestion, or incentive in the prior art, *Kandasamy* cannot be properly modified using the teachings of *Chrabaszcz* to form the claimed invention. The motivation suggested in the Office Action of "providing multiple services from different servers for clients which increases scalability" is a general motivation that does not necessarily lead a person of ordinary skill in the art to make the modifications proposed in the Office Action. The Office Action does not explain why a person of ordinary skill in the art would be driven to make the specific modifications that would be necessary to arrive at the claimed invention.

If an independent claim is non-obvious under 35 U.S.C. § 103, then any claim depending therefrom is non-obvious. *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). Claims 10 and 12 are dependent claims that depend on independent claims 9 and 11, respectively. Applicants have already demonstrated claims 9 and 11 to be in condition for allowance. Applicants respectfully submit that claims 10 and 12 are also allowable, at least by virtue of their dependency on allowable claims. In addition, these dependent claims recite additional subject matter not taught or suggested by the cited references. For example, claim 12 recites removing the second server name prior to connecting the second server to a network in the distributed data processing system, which is neither taught nor suggested by the prior art of record.

For the foregoing reasons, Applicants submit that claims 9-12 and 23 are patentable over the references. Accordingly, Applicants respectfully request that the rejection of claims 9-12 and 23 be withdrawn.

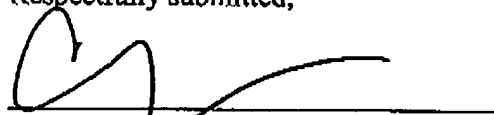
**III. Conclusion**

It is respectfully urged that the subject application is patentable over the cited references and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: 7/16/03

Respectfully submitted,



Cathrine K. Kinslow  
Reg. No. 51,886  
Carstens, Yee & Cahoon, LLP  
P.O. Box 802334  
Dallas, TX 75380  
(972) 367-2001  
Attorney for Applicants